Introduction to MINLP

Ralf Lenz

Zuse Institute Berlin

$$
\begin{aligned}
& \min f(x) \\
\text{s.t.} \quad & g_k(x) \leqslant 0 && k = 1, \ldots, m, \\
& x \in X = \{x \in \mathbb{R}^n : Dx \leq d, x \in [\ell, u]\} \\
& x_i \in \mathbb{Z} && i \in \mathcal{I} \subseteq \{1, \ldots, n\}.
\end{aligned}
$$

$$
\begin{aligned}
&\min f(x) \\
\text{s.t.}\quad & g_k(x) \leqslant 0 && k = 1, \dots, m, \\
& x \in X = \{x \in \mathbb{R}^n : Dx \leq d, x \in [\ell, u]\} \\
& x_i \in \mathbb{Z} && i \in \mathcal{I} \subseteq \{1, \dots, n\}.
\end{aligned}
$$

in case of a nonlinear objective function $f(x)$:

$$
\begin{aligned}
&\min x_0, \\
\text{s.t.}\quad & g_k(x) \leqslant 0 && k = 1, \dots, m, \\
& f(x) - x_0 \leqslant 0 \\
& x \in X = \{x \in \mathbb{R}^n : Dx \leq d, x \in [\ell, u]\} \\
& x_i \in \mathbb{Z} && i \in \mathcal{I} \subseteq \{1, \dots, n\}.
\end{aligned}
$$

$$
\begin{aligned}
&\min f(x) && \\
\text{s.t.}\quad & g_k(x) \leqslant 0 && k = 1, \dots, m, \\
& x \in X = \{x \in \mathbb{R}^n : Dx \leq d, x \in [\ell, u]\} && \\
& x_i \in \mathbb{Z} && i \in \mathcal{I} \subseteq \{1, \dots, n\}.
\end{aligned}
$$

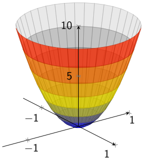
Special cases:

- $m = 0$ and $\mathcal{I} \neq \emptyset \Rightarrow$ Mixed Integer Program (MIP)
  $\rightsquigarrow$ lectures by Bob Bixby, Tobias Achterberg next Monday
- $\mathcal{I} = \emptyset \Rightarrow$ NonLinear Program (NLP)

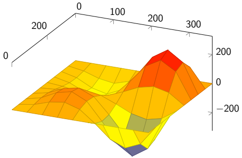
$$
\begin{aligned}
&\min f(x) \\
\text{s.t.}\quad & g_k(x) \leqslant 0 && k = 1, \ldots, m, \\
& x \in X = \{x \in \mathbb{R}^n : Dx \leq d, x \in [\ell, u]\} \\
& x_i \in \mathbb{Z} && i \in \mathcal{I} \subseteq \{1, \ldots, n\}.
\end{aligned}
$$

Special cases:

▷ convex or nonconvex MINLP depending on the (non)convexity of f, g_k



convex
local = global optimality



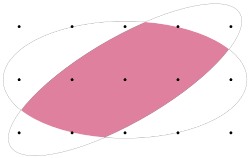
nonconvex
suboptimal local optima

$\rightsquigarrow$ talk by Pierre Bonami next Thursday

Assumption $g_1, \dots, g_m$ convex.

MINLP

$$\begin{aligned} &\min x_0 \\ s.t.\ & g_k(x) \leqslant 0,\ k \in [m] \\ & x \in X \\ & x_i \in \mathbb{Z},\ i \in \mathcal{I} \end{aligned}$$



$\{x \in X \mid g_k(x) \leq 0\ \forall k, x_i \in \mathbb{Z},\ i \in \mathcal{I}\}$

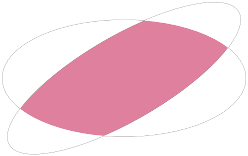
Assumption $g_1, \dots, g_m$ convex.

MINLP

$$\begin{aligned} & \min x_0 \\ s.t.\ & g_k(x) \leqslant 0,\ k \in [m] \\ & x \in X \\ & x_i \in \mathbb{Z},\ i \in \mathcal{I} \end{aligned}$$

NLP Relaxation

$$\begin{aligned} & \min x_0 \\ s.t.\ & g_k(x) \leqslant 0,\ k \in [m] \\ & x \in X \end{aligned}$$

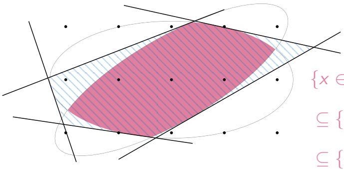


$\{x \in X \mid g_k(x) \leq 0\ \forall k, x_i \in \mathbb{Z},\ i \in \mathcal{I}\}$

$\subseteq \{x \in X \mid Ax \leq b, x_i \in \mathbb{Z},\ i \in \mathcal{I}\}$

Assumption $g_1, \dots, g_m$ convex.

MINLP	MIP Relaxation
$\min x_0$	$\min\ x_0$
$s.t.\ g_k(x) \leqslant 0,\ k \in [m]$	$s.t.\ Ax \leq b$
$x \in X$	$x \in X$
$x_i \in \mathbb{Z},\ i \in \mathcal{I}$	$x_i \in \mathbb{Z},\ i \in \mathcal{I}$



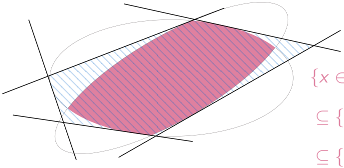
$\{x \in X \mid g_k(x) \leq 0\ \forall k, x_i \in \mathbb{Z},\ i \in \mathcal{I}\}$

$\subseteq \{x \in X \mid Ax \leq b, x_i \in \mathbb{Z},\ i \in \mathcal{I}\}$

$\subseteq \{x \in X \mid Ax \leq b\}$

Assumption $g_1, \ldots, g_m$ convex.

MINLP	MIP Relaxation	LP Relaxation
$\begin{aligned} & \min x_0 \\ s.t.\ & g_k(x) \leqslant 0,\ k \in [m] \\ & x \in X \\ & x_i \in \mathbb{Z},\ i \in \mathcal{I} \end{aligned}$	$\begin{aligned} & \min\ x_0 \\ s.t.\ & Ax \leq b \\ & x \in X \\ & x_i \in \mathbb{Z},\ i \in \mathcal{I} \end{aligned}$	$\begin{aligned} & \min\ x_0 \\ s.t.\ & Ax \leq b \\ & x \in X \end{aligned}$



$\{x \in X \mid g_k(x) \leq 0\ \forall k, x_i \in \mathbb{Z},\ i \in \mathcal{I}\}$

$\subseteq \{x \in X \mid Ax \leq b, x_i \in \mathbb{Z},\ i \in \mathcal{I}\}$

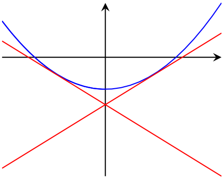
$\subseteq \{x \in X \mid Ax \leq b\}$

How to construct the LP relaxation?

Assumption g_k are differentiable and convex.

For simplicity consider $m = 1$:

$$\text{Then } x^j \in [\ell, u] : \quad \underbrace{g(x^j) + \nabla g(x^j)^\mathsf{T}(x - x^j)}_{=: L^j(x)} \leqslant g(x) \ \forall x \in [\ell, u]$$

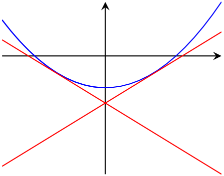


How to construct the LP relaxation?

Assumption g_k are differentiable and convex.

For simplicity consider $m = 1$:

$$\text{Then } x^j \in [\ell, u]: \quad \underbrace{g(x^j) + \nabla g(x^j)^\mathsf{T}(x - x^j)}_{=:L^j(x)} \leqslant g(x)\ \forall x \in [\ell, u]$$



Proposition

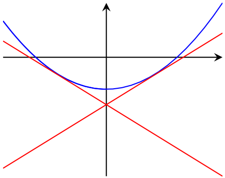
Consider $\{x^j \in [\ell, u] \mid j \in [r]\}$ *and replace* $g(x) \leq 0$ *by* $L^j(x) \leq 0$. *Then* $\cap_i \{x \mid L^j(x) \leq 0\}$ *is a polyhedral relaxation of* $g(x) \leq 0$.

How to construct the LP relaxation?

Assumption g_k are differentiable and convex.

For simplicity consider $m = 1$:

$$\text{Then } x^j \in [\ell, u] : \quad \underbrace{g(x^j) + \nabla g(x^j)^\mathsf{T}(x - x^j)}_{=:L^j(x)} \leqslant g(x)\ \forall x \in [\ell, u]$$



Let $\mathcal{J} \subset \mathbb{R}^n$ be a finite set of points, then the LP Relaxation is given by:

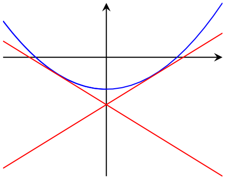
$$\begin{aligned} &\min\ x_0 \\ s.t.\ & g_k(x^j) + \nabla g_k(x^j)^\mathsf{T}(x - x^j) \leq 0, \quad x^j \in \mathcal{J}, \forall k \in [m] \\ & x \in X \end{aligned}$$

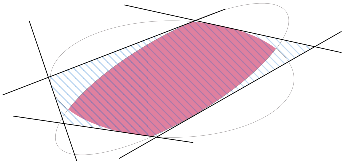
How to construct the LP relaxation?

Assumption g_k are differentiable and convex.

For simplicity consider $m = 1$:

$$\text{Then } x^j \in [\ell, u]: \quad \underbrace{g(x^j) + \nabla g(x^j)^\mathsf{T}(x - x^j)}_{=:L^j(x)} \leqslant g(x)\ \forall x \in [\ell, u]$$

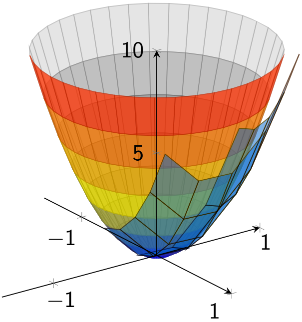




Let $\mathcal{J} \subset \mathbb{R}^n$ be a finite set of points, then the LP Relaxation is given by:

$$\begin{aligned} &\min\ x_0 \\ s.t.\ & g_k(x^j) + \nabla g_k(x^j)^\mathsf{T}(x - x^j) \leq 0, \quad x^j \in \mathcal{J}, \forall k \in [m] \\ & x \in X \end{aligned}$$

Polyhedral relaxation: take advantage of convexity and apply gradient cuts to get linear outer approximation

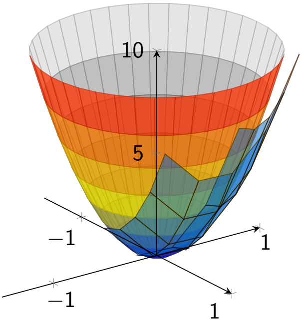


Polyhedral relaxation: take advantage of convexity and apply gradient cuts to get linear outer approximation

Solving methods

Branch and Bound

- LP based
  - solve LP relaxations

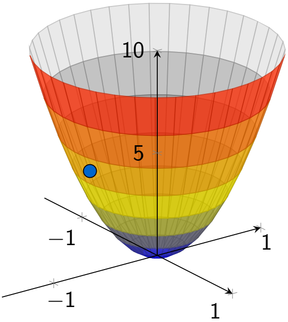


Polyhedral relaxation: take advantage of convexity and apply gradient cuts to get linear outer approximation

Solving methods

Branch and Bound

- LP based
  - solve LP relaxations
- NLP-based
  - solve NLP relaxations
- relaxation $\Rightarrow$ lower bound at each node
- branch on fractional integer variables

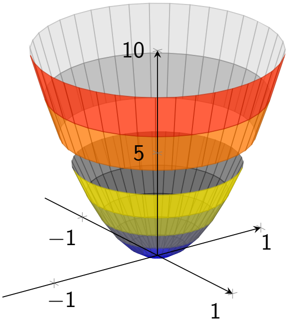


Polyhedral relaxation: take advantage of convexity and apply gradient cuts to get linear outer approximation

Solving methods

Branch and Bound

- LP based
  - solve LP relaxations
- NLP-based
  - solve NLP relaxations
- relaxation ⇒ lower bound at each node
- branch on fractional integer variables

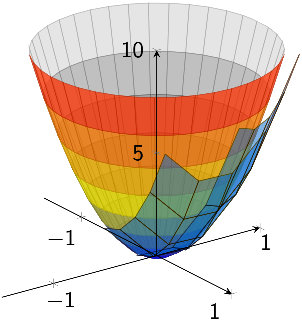


Polyhedral relaxation: take advantage of convexity and apply gradient cuts to get linear outer approximation

Solving methods

Branch and Bound

- ▷ LP based
  - ▷ solve LP relaxations
- ▷ NLP-based
  - ▷ solve NLP relaxations
- ▷ relaxation $\Rightarrow$ lower bound at each node
- ▷ branch on fractional integer variables



Outer Approximation

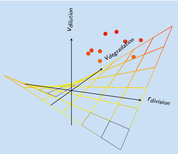
$\rightsquigarrow$ different solving methods

Applications, applications, applications

- industrial engineering: mining with stockpiling constraints
- networks: operation and design of water and gas networks
- energy production and distribution: plant design, power scheduling
- chemical industry: design of synthesis processes
- ...



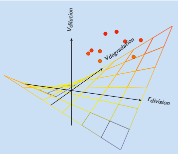




Applications, applications, applications

- industrial engineering: mining with stockpiling constraints
- networks: operation and design of water and gas networks
- energy production and distribution: plant design, power scheduling
- chemical industry: design of synthesis processes
- ...

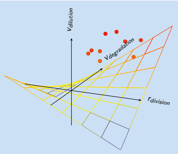


Applications, applications, applications

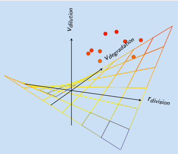
- industrial engineering: mining with stockpiling constraints
- networks: operation and design of water and gas networks
- energy production and distribution: plant design, power scheduling
- chemical industry: design of synthesis processes
- ...



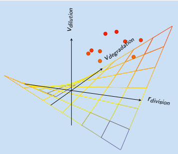


Applications, applications, applications

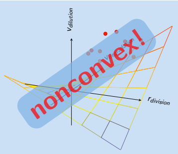
- ▷ industrial engineering: mining with stockpiling constraints
- ▷ networks: operation and design of water and gas networks
- ▷ energy production and distribution: plant design, power scheduling
- ▷ chemical industry: design of synthesis processes
- ▷ ...

- industrial engineering: mining with stockpiling constraints
- networks: operation and design of water and gas networks
- energy production and distribution: plant design, power scheduling
- chemical industry: design of synthesis processes
- . . .

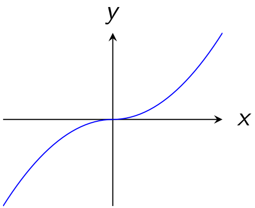


- industrial engineering: mining with stockpiling constraints
- networks: operation and design of water and gas networks
- energy production and distribution: plant design, power scheduling
- chemical industry: design of synthesis processes
- ...



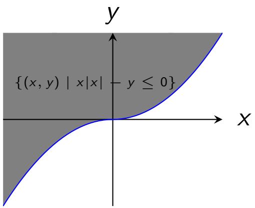
Example: Assume g to be nonconvex

$$g(x, y) = x|x| - y = 0$$



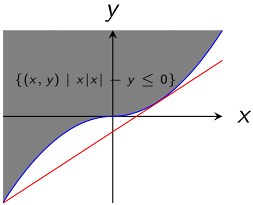
Example: Assume g to be nonconvex

$$g(x, y) = x|x| - y \leq 0$$



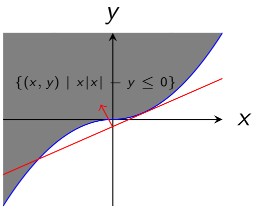
Example: Assume g to be nonconvex

$$g(x, y) = x|x| - y \leq 0$$



Example: Assume g to be nonconvex

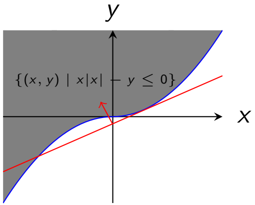
$$g(x, y) = x|x| - y \leq 0$$



Gradient cuts may not be valid!

Example: Assume g to be nonconvex

$$g(x, y) = x|x| - y \leq 0$$



Gradient cuts may not be valid!

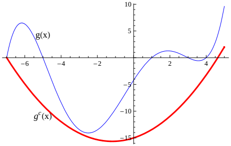
The following difficulties occur in nonconvex problems:

- How to generate the outer approximation?
- How to enforce nonlinear constraints?

How to generate the outer approximation?

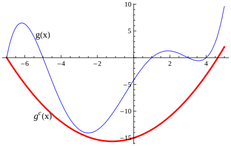
How to generate the outer approximation?

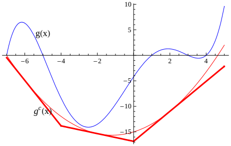
▷ use convex underestimator instead (= convex function below $g(x)$ $\forall x \in [\ell, u]$)



How to generate the outer approximation?

- ▷ use convex underestimator instead ($=$ convex function below $g(x)$ $\forall x \in [\ell, u]$)

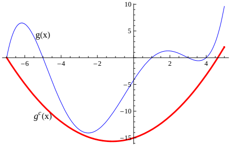


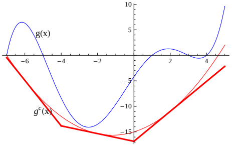


- ▷ compute gradient cuts to the convex underestimator

How to generate the outer approximation?

- use convex underestimator instead (= convex function below $g(x)\ \forall x \in [\ell, u]$)

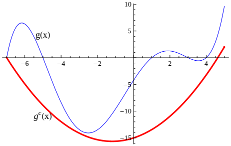


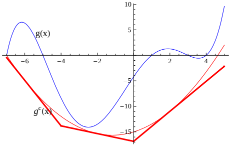


- compute gradient cuts to the convex underestimator
- convex envelopes (= tightest convex underestimator)

How to generate the outer approximation?

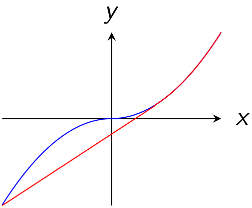
- ▷ use convex underestimator instead (= convex function below $g(x)$ $\forall x \in [\ell, u]$)





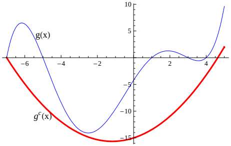
- ▷ compute gradient cuts to the convex underestimator
- ▷ convex envelopes (= tightest convex underestimator)

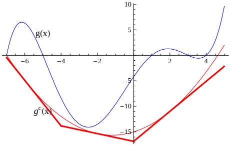
Example: convex envelope



How to generate the outer approximation?

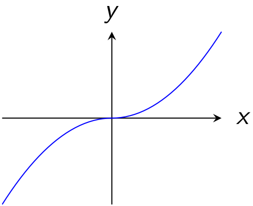
- use convex underestimator instead (= convex function below $g(x)$ $\forall x \in [\ell, u]$)





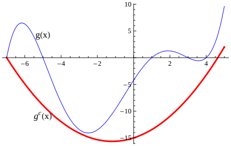
- compute gradient cuts to the convex underestimator
- convex envelopes (= tightest convex underestimator)

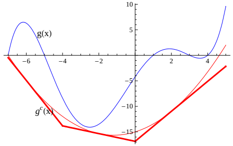
Example: convex envelope and concave envelope for $g(x, y) = x|x| - y = 0$



How to generate the outer approximation?

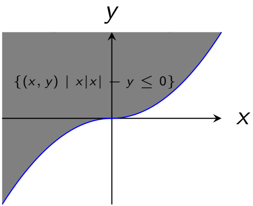
- use convex underestimator instead (= convex function below $g(x)$ $\forall x \in [\ell, u]$)





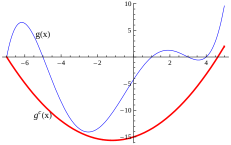
- compute gradient cuts to the convex underestimator
- convex envelopes (= tightest convex underestimator)

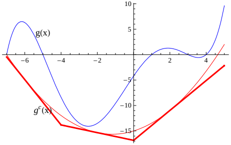
Example: convex envelope and concave envelope for $g(x, y) = x|x| - y = 0$



How to generate the outer approximation?

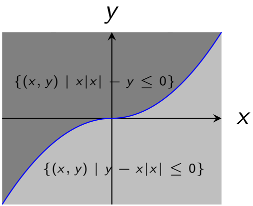
- use convex underestimator instead (= convex function below $g(x)$ $\forall x \in [\ell, u]$)





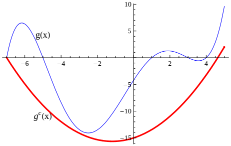
- compute gradient cuts to the convex underestimator
- convex envelopes (= tightest convex underestimator)

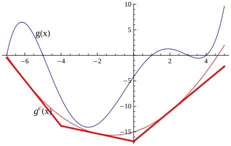
Example: convex envelope and concave envelope for $g(x, y) = x|x| - y = 0$



How to generate the outer approximation?

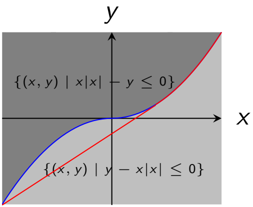
- use convex underestimator instead (= convex function below $g(x)$ $\forall x \in [\ell, u]$)





- compute gradient cuts to the convex underestimator
- convex envelopes (= tightest convex underestimator)

Example: convex envelope and concave envelope for $g(x, y) = x|x| - y = 0$

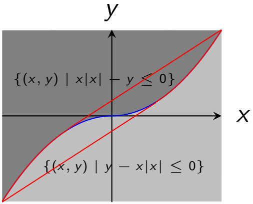


How to generate the outer approximation?

- ▷ use convex underestimator instead (= convex function below $g(x)$ $\forall x \in [\ell, u]$)

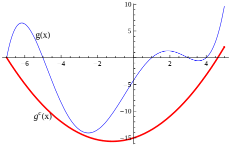
- ▷ compute gradient cuts to the convex underestimator
- ▷ convex envelopes (= tightest convex underestimator)

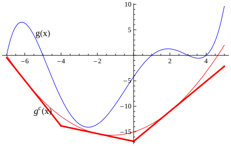
Example: convex envelope and concave envelope for $g(x, y) = x|x| - y = 0$



How to generate the outer approximation?

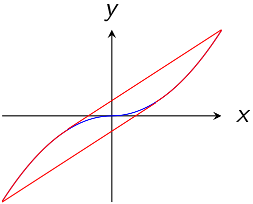
- use convex underestimator instead (= convex function below $g(x)\ \forall x \in [\ell, u]$)





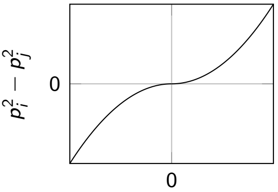
- compute gradient cuts to the convex underestimator
- convex envelopes (= tightest convex underestimator)

Example: convex envelope and concave envelope for $g(x, y) = x|x| - y = 0$



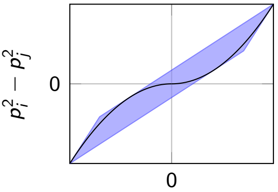
How to enforce nonlinear constraints? Branching on continuous variables

- Outer approximation



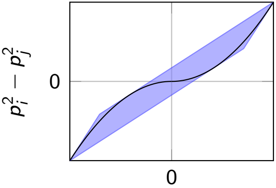
How to enforce nonlinear constraints? Branching on continuous variables

- Outer approximation



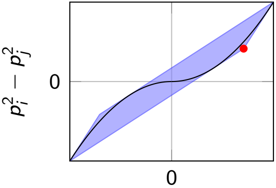
How to enforce nonlinear constraints? Branching on continuous variables

- Outer approximation



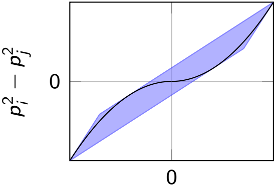
- Cutting planes

$$g_k(\hat{x}) + \nabla g_k(\hat{x})^\mathsf{T}(x - \hat{x}) \leqslant 0$$



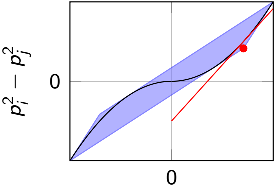
How to enforce nonlinear constraints? Branching on continuous variables

- Outer approximation



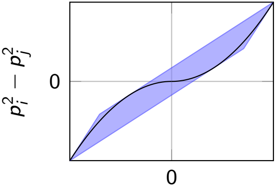
- Cutting planes

$$g_k(\hat{x}) + \nabla g_k(\hat{x})^\mathsf{T}(x - \hat{x}) \leqslant 0$$



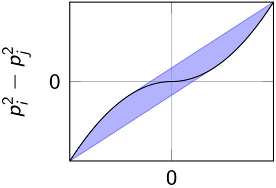
How to enforce nonlinear constraints? Branching on continuous variables

- Outer approximation



- Cutting planes

$$g_k(\hat{x}) + \nabla g_k(\hat{x})^\mathsf{T}(x - \hat{x}) \leqslant 0$$



How to enforce nonlinear constraints? Branching on continuous variables

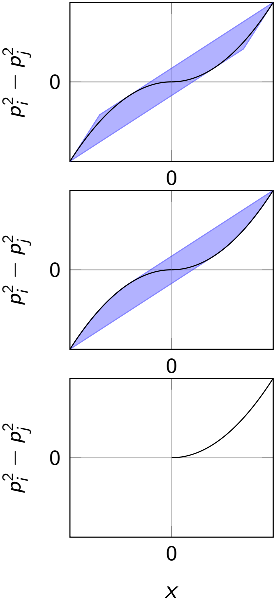
- Outer approximation

- Cutting planes

  $$g_k(\hat{x}) + \nabla g_k(\hat{x})^{\mathsf{T}}(x - \hat{x}) \leqslant 0$$

- Spatial Branching

  $x \geq 0$



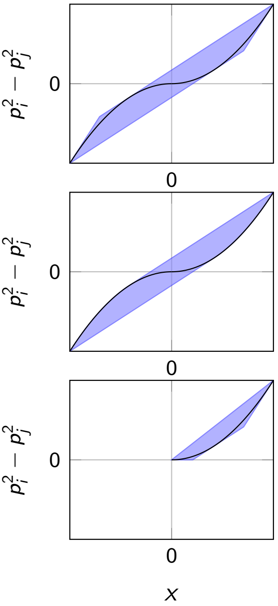
How to enforce nonlinear constraints? Branching on continuous variables

- Outer approximation
- Cutting planes

  $$g_k(\hat{x}) + \nabla g_k(\hat{x})^{\mathsf{T}}(x - \hat{x}) \leqslant 0$$

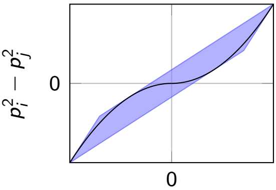
- Spatial Branching

  $x \geq 0$



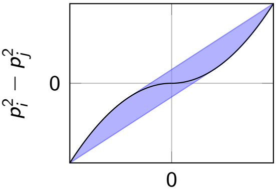
How to enforce nonlinear constraints? Branching on continuous variables

- Outer approximation



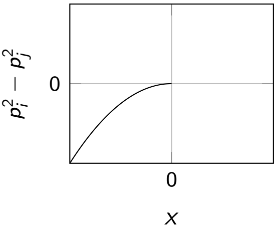
- Cutting planes

$$g_k(\hat{x}) + \nabla g_k(\hat{x})^\mathsf{T}(x - \hat{x}) \leqslant 0$$



- Spatial Branching

$x \leq 0$



Nonconvex MINLP - Spatial Branching

How to enforce nonlinear constraints? Branching on continuous variables

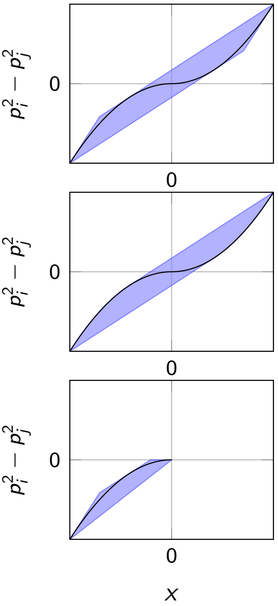
- Outer approximation

- Cutting planes

  $$g_k(\hat{x}) + \nabla g_k(\hat{x})^\mathsf{T}(x - \hat{x}) \leqslant 0$$

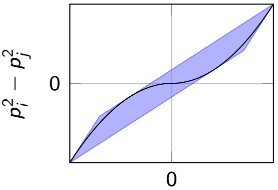
- Spatial Branching

  $$x \leq 0$$



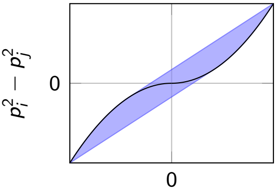
How to enforce nonlinear constraints? Branching on continuous variables

- Outer approximation



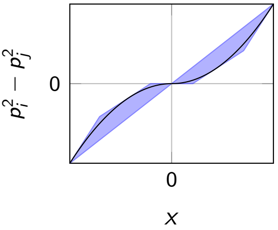
- Cutting planes

$$g_k(\hat{x}) + \nabla g_k(\hat{x})^{\mathsf{T}}(x - \hat{x}) \leqslant 0$$



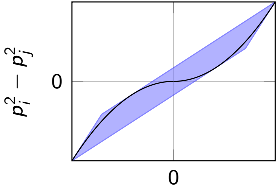
- Spatial Branching

$x \leq 0$ and $x \geq 0$



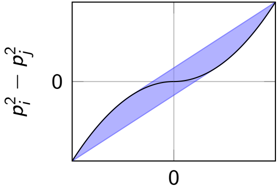
How to enforce nonlinear constraints? Branching on continuous variables

- Outer approximation

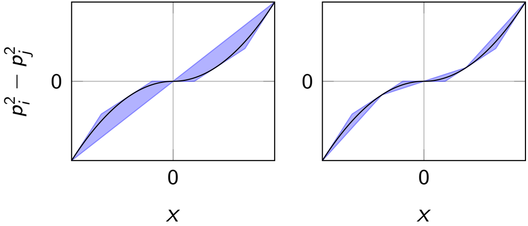


- Cutting planes

$$g_k(\hat{x}) + \nabla g_k(\hat{x})^\mathsf{T}(x - \hat{x}) \leqslant 0$$



- Spatial Branching



SCIP is an LP based Branch & Bound solver

In each node:

SCIP is an LP based Branch & Bound solver

In each node:

- **Bounding:** solve LP relaxation

$$\begin{aligned} &\min\ x_0 \\ s.t.\ &g_k(x^j) + \nabla g_k(x^j)^{\mathsf{T}}(x - x^j) \leq 0,\ x^j \in \mathcal{J}, \forall k \in [m] \\ &x \in X \end{aligned}$$

SCIP is an LP based Branch & Bound solver

In each node:

- Bounding: solve LP relaxation

$$\begin{aligned} &\min\ x_0 \\ s.t.\ &g_k(x^j) + \nabla g_k(x^j)^\mathsf{T}(x - x^j) \leq 0,\ x^j \in \mathcal{J}, \forall k \in [m] \\ &x \in X \end{aligned}$$

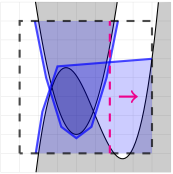
- Strengthen relaxation with
  - Cutting planes

SCIP is an LP based Branch & Bound solver

In each node:

- Bounding: solve LP relaxation

$$
\begin{aligned}
&\min\ x_0\\
s.t.\ &g_k(x^j) + \nabla g_k(x^j)^{\mathsf{T}}(x - x^j) \le 0,\ x^j \in \mathcal{J}, \forall k \in [m]\\
&x \in X
\end{aligned}
$$



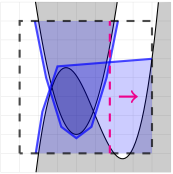
- Strengthen relaxation with
  - Cutting planes
  - Bound tightening

SCIP is an LP based Branch & Bound solver

In each node:

- Bounding: solve LP relaxation

$$\begin{aligned} &\min\ x_0 \\ s.t.\ &g_k(x^j) + \nabla g_k(x^j)^\mathsf{T}(x - x^j) \leq 0,\ x^j \in \mathcal{J}, \forall k \in [m] \\ &x \in X \end{aligned}$$



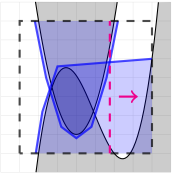
- Strengthen relaxation with
  - Cutting planes
  - Bound tightening
- Primal heuristics to search for feasible solutions

SCIP is an LP based Branch & Bound solver

In each node:

- Bounding: solve LP relaxation

$$\begin{aligned} &\min\ x_0 \\ s.t.\ & g_k(x^j) + \nabla g_k(x^j)^\mathsf{T}(x - x^j) \le 0,\ x^j \in \mathcal{J}, \forall k \in [m] \\ & x \in X \end{aligned}$$



- Strengthen relaxation with
  - Cutting planes
  - Bound tightening
- Primal heuristics to search for feasible solutions
- Branching
  - at first on fractional integer variables
  - then apply Spatial Branching